**Amendments to the claims,**
**Listing of all claims pursuant to 37 CFR 1.121(c)**

*This listing of claims will replace all prior versions, and listings, of claims in the application:*

What is claimed is:

1. (Currently amended) In a database system, a method for providing automated encryption support for column data, the method comprising:

defining Structured Query Language (SQL) extensions for creating and managing column encryption keys, and for creating and managing database tables with encrypted column data;

receiving ~~an~~ a first SQL statement ~~specifying creation of~~ employing said SQL extensions to create a named encryption key for encrypting column data~~, said named encryption key capable of encrypting multiple columns~~;

parsing the first SQL statement, including creating said named encryption key with a syntactically unique name that can be parsed from within other SQL statements employing said SQL extensions;

receiving ~~at least one~~ a second SQL statement ~~specifying creation of~~ employing said SQL extensions to create a database table ~~having encrypted column data, each such SQL statement specifying a database table~~ having particular column data encrypted with said named encryption key;

parsing the second SQL statement, including identifying said named encryption key upon parsing a portion of the statement that comprises the syntactically unique name for the key;

in response to parsing the second SQL statement, creating a database table having particular column data encrypted with said named encryption key identified upon parsing the second SQL statement; and

in response to a subsequent database operation that requires particular column data that has been encrypted with said named encryption key, automatically decrypting the particular column data with said named encryption key, so that the particular column data is available in decrypted form for use by the database operation.

2. (Original) The method of claim 1, wherein columns that are not specified to be encrypted are stored in unencrypted format, for minimizing encryption overhead.

3. (Original) The method of claim 1, wherein the automated encryption support operates as an internal built-in feature of the database system, without use of an add-on library.

4. (Currently amended) The method of claim 1, wherein ~~the~~ said first SQL statement ~~specifying creation of a named encryption key~~ employing said SQL extensions to create a named encryption key is received from a user serving as a system security officer.

5. (Currently amended) The method of claim 4, wherein ~~the~~ said second SQL statement ~~specifying creation of a database table~~ employing said SQL extensions to create a database table having particular column data encrypted with said named encryption key may be received from a user other than the system security officer.

6. (Currently amended) The method of claim 1, wherein ~~the~~ said first SQL statement ~~specifying creation of a named encryption key comprises~~ employs an SQL extension having a CREATE ENCRYPTION KEY command.

7. (Original) The method of claim 6, wherein the CREATE ENCRYPTION KEY command includes:

```
CREATE ENCRYPTION KEY keyname
   [AS DEFAULT] [FOR algorithm]
   [WITH [KEYLENGTH keysize]
      [PASSWD passphrase]
      [INIT_VECTOR [RANDOM | NULL]]
      [PAD [RANDOM | NULL]]]
```

as its syntax.

8. (Currently amended) The method of claim 1, wherein ~~the at least one~~ said second SQL statement ~~specifying creation of a database table having particular column data encrypted~~ comprises a CREATE TABLE command that allows specification of one or more columns to be encrypted.

9. (Currently amended) The method of claim 8, wherein the CREATE TABLE command includes:

```
CREATE TABLE tablename
  (colname1 datatype [encrypt [with [db.[owner].]keyname]],
  colname2 datatype [encrypt [with [db.[owner].]keyname]])
```

as its syntax.

10. (Original) The method of claim 1, further comprising:

receiving an SQL statement specifying alteration of a previously-created database table so as to encrypt particular column data.

11. (Original) The method of claim 10, wherein the SQL statement specifying alteration of a previously created database table comprises an ALTER TABLE command.

12. (Original) The method of claim 11, wherein the ALTER TABLE command includes:

```
ALTER TABLE tablename MODIFY column_name
  [[datatype] [null|not null]]
  [decrypt | encrypt [with [db.[owner].]keyname]]
```

as its syntax.

13. (Original) The method of claim 1, wherein the encryption support works transparently with existing database applications.

14. (Original) The method of claim 1, wherein the database system includes a database server and one or more database clients, and wherein method steps

implementing the encryption support are embodied at the database server.

15. (Original) The method of claim 1, wherein the database system includes a back-end server tier and a middleware tier, and wherein method steps implementing the encryption support are embodied at the back-end server tier.

16. (Previously presented) The method of claim 1, further comprising:

after creation of the named encryption key, protecting the named encryption key with a user-supplied password.

17. (Previously presented) The method of claim 16, wherein the user-supplied password must be supplied before the system allows use of the named encryption key for database operations.

18. (Original) The method of claim 17, wherein the user-supplied password is supplied using a SET ENCRYPTION PASSWD command.

19. (Original) The method of claim 18, wherein the SET ENCRYPTION PASSWD command includes:

SET ENCRYPTION PASSWD password FOR keyname

as its syntax.

20. (Previously presented) The method of claim 17, wherein a user seeking to decrypt column data must supply said user-supplied password and must have necessary database privileges before decrypting the column data with the named encryption key.

21. (Original) The method of claim 20, wherein the user-supplied password is supplied using a SET ENCRYPTION PASSWD command.

22. (Original) The method of claim 1, further comprising:

providing a command to grant decryption permission to others.

23. (Original) The method of claim 22, wherein the command to grant decryption permission includes:

GRANT DECRYPT ON table.column TO user_or_role_list

as its syntax.

24. (Original) The method of claim 1, wherein the database system internally stores in encrypted format any column encryption keys that have been created.

25. (Original) The method of claim 1, wherein the database system stores encrypted column data internally as variable binary (VARBINARY) data.

26. (Original) The method of claim 1, wherein the database system presents users a user-defined field type for column data that has been encrypted, even though the column data is stored internally as variable binary data.

27. (Original) The method of claim 1, wherein the database system preserves any user-defined data type for the particular column data so that the database system employs a correct data type for processing queries and returning query results.

28. (Original) The method of claim 27, wherein the database system stores the user-defined data type for the particular column data in a system catalog of the database system.

29. (Previously presented) The method of claim 1, wherein the named encryption key created comprises a symmetric encryption key.

30. (Previously presented) The method of claim 1, wherein a single column named encryption key is used for each column to be encrypted.

31. (Original) The method of claim 1, wherein a single column encryption key

may be shared by multiple columns to be encrypted.

32. (Currently amended) The method of claim 1, wherein the named encryption key is itself encrypted ~~to~~ by a key-encrypting key constructed from a user-supplied password.

33. (Previously presented) The method of claim 32, wherein the named encryption key is itself stored on disk in encrypted format using Advanced Encryption Standard (AES) encryption.

34. (Original) The method of claim 32, wherein the user-supplied password may comprise a hex literal.

35. (Original) The method of claim 32, wherein the user-supplied password is itself transformed into a symmetric encryption key, using a random salt, internal static data, and SHA-1 hashing algorithm.

36. (Previously presented) The method of claim 1, wherein said Structured Query Language (SQL) extensions for creating and managing named encryption keys include a clause for instructing the database system to create a default key for encrypting columns.

37. (Currently amended) A database system providing automated encryption support for column data, the system comprising:

a processor;

a memory coupled to the processor;

a parser that supports Structured Query Language (SQL) extensions for creating and managing named ~~column~~ encryption keys for encrypting column data, and for creating and managing database tables with encrypted column data; and

an execution unit, operating in response to SQL statements parsed by the parser, that:

~~for creating~~ creates in response to parsing a first SQL statement employing said SQL extensions a particular named ~~column~~ encryption key having a syntactically unique name that can be parsed from within other SQL statements employing said SQL extensions,

~~for creating~~ creates in response to parsing a second SQL statement employing said SQL extensions one or more database tables having particular column data encrypted with said particular named ~~column~~ encryption key, including identifying said particular named encryption key upon parsing a portion of the statement that comprises the syntactically unique name for the key, and

~~for automatically decrypting~~ automatically decrypts the particular column data for use by a subsequent database operation that requires the particular column data that has been encrypted.

38. (Original) The system of claim 37, wherein columns that are not specified to be encrypted are stored in unencrypted format, for minimizing encryption overhead.

39. (Original) The system of claim 37, wherein the automated encryption support operates as an internal built-in feature of the database system, without use of an add-on library.

40. (Currently amended) The system of claim 37, wherein ~~the~~ said first SQL statement specifying creation of a particular named encryption key is received from a user serving as a system security officer.

41. (Currently amended) The system of claim 40, wherein ~~the~~ said second SQL statement specifying creation of one or more database tables may be received from a user other than the system security officer.

42. (Currently amended) The system of claim 37, wherein ~~the~~ said first SQL statement specifying creation of a particular named encryption key comprises a CREATE ENCRYPTION KEY command.

43. (Original) The system of claim 42, wherein the CREATE ENCRYPTION KEY command includes:

```
CREATE ENCRYPTION KEY keyname
   [AS DEFAULT] [FOR algorithm]
   [WITH [KEYLENGTH keysize]
      [PASSWD passphrase]
      [INIT_VECTOR [RANDOM | NULL]]
      [PAD [RANDOM | NULL]]]
```

as its syntax.

44. (Currently amended) The system of claim 37, wherein ~~the~~ said second SQL statement specifying creation of one or more database tables having particular column data encrypted comprises a CREATE TABLE command that allows specification of one or more columns to be encrypted.

45. (Original) The system of claim 44, wherein the CREATE TABLE command includes:

```
CREATE TABLE tablename
  (colname1 datatype [encrypt [with [db.[owner].]keyname],
  colname2 datatype [encrypt [with [db.[owner].]keyname])
```

as its syntax.

46. (Original) The system of claim 37, further comprising:

a module for receiving an SQL statement specifying alteration of a previously created database table so as to encrypt particular column data.

47. (Original) The system of claim 46, wherein the SQL statement specifying alteration of a previously created database table comprises an ALTER TABLE command.

48. (Original) The system of claim 47, wherein the ALTER TABLE command includes:

```
ALTER TABLE tablename MODIFY column_name
  [[datatype] [null|not null]]
  [decrypt | encrypt [with [db.[owner].]keyname]]
```
as its syntax.

49. (Original) The system of claim 37, wherein the encryption support works transparently with existing database applications.

50. (Original) The system of claim 37, wherein the database system includes a database server and one or more database clients, and wherein the encryption support is provided by the database server.

51. (Original) The system of claim 37, wherein the database system includes a back-end server tier and a middleware tier, and wherein the encryption support is provided by the back-end server tier.

52. (Currently amended) The system of claim 37, wherein the system protects the particular named ~~column~~ encryption key with a user-supplied password.

53. (Currently amended) The system of claim 52, wherein the user-supplied password must be supplied before the system allows use of the particular named ~~column~~ encryption key for database operations.

54. (Original) The system of claim 53, wherein the user-supplied password is supplied using a SET ENCRYPTION PASSWD command.

55. (Original) The system of claim 54, wherein the SET ENCRYPTION PASSWD command includes:

```
SET ENCRYPTION PASSWD password FOR keyname
```
as its syntax.

56. (Currently amended) The system of claim 53, wherein a user seeking to decrypt column data must supply said user-supplied password and must have necessary database privileges before decrypting the column data with the particular named ~~column~~ encryption key.

57. (Original) The system of claim 37, further comprising:

providing a command to grant decryption permission to others.

58. (Original) The system of claim 57, wherein the command to grant decryption permission includes:

GRANT DECRYPT ON table.column TO user_or_role_list

as its syntax.

59. (Currently amended) The system of claim 37, wherein the database system internally stores in encrypted format any named ~~column~~ encryption keys that have been created.

60. (Original) The system of claim 37, wherein the database system stores encrypted column data internally as variable binary (VARBINARY) data.

61. (Original) The system of claim 37, wherein the database system presents users a user-defined field type for column data that has been encrypted, even though the column data is stored internally as variable binary data.

62. (Original) The system of claim 37, wherein the database system preserves any user-defined data type for the particular column data so that the database system employs a correct data type for processing queries and returning query results.

63. (Original) The system of claim 62, wherein the database system stores the user-defined data type for the particular column data in a system catalog of the database system.

64. (Currently amended) The system of claim 37, wherein the particular named ~~column~~ encryption key created comprises a symmetric encryption key.

65. (Previously presented) The system of claim 37, wherein a single column named encryption key is used for each column to be encrypted.

66. (Currently amended) The system of claim 37, wherein the particular named ~~column~~ encryption key is itself encrypted ~~to~~ by a key-encrypting key constructed from a user-supplied password.

67. (Currently amended) The system of claim 66, wherein the particular named ~~column~~ encryption key is itself stored on disk in encrypted format using Advanced Encryption Standard (AES) encryption.

68. (Original) The system of claim 66, wherein the user-supplied password may comprise a hex literal.

69. (Original) The system of claim 66, wherein the user-supplied password is itself transformed into a symmetric encryption key, using a random salt, static internal data and SHA-1 hashing algorithm.

70. (Currently amended) The system of claim 37, wherein said Structured Query Language (SQL) extensions for creating and managing named ~~column~~ encryption keys include a clause for instructing the database system to create a default key for encrypting columns.

71. (Currently amended) In a database system, a method for encrypting column data, the method comprising:

defining query language extensions for creating and managing column encryption keys, and for creating and managing database tables with encrypted column data;

in response to a first query language statement employing said extensions, creating a named encryption key for encrypting a particular column of a database table, said named encryption key being ~~uniquely named~~ created with a syntactically unique name so that it can be referenced within other query language statements employing said extensions;

in response to a second query language statement employing said extensions, encrypting the particular column using said named encryption key, ~~said second query language statement referencing said named encryption key by its unique name~~ including identifying said named encryption key upon parsing a portion of the statement that comprises the syntactically unique name for the key; and

during a subsequent database operation requiring column data inserted to or selected from the particular column, automatically encrypting or decrypting the column data as necessary for carrying out the database operation.

72. (Original) The method of claim 71, further comprising:

assigning privileges to users for creating an encryption key for encrypting column data.

73. (Previously presented) The method of claim 72, further comprising:

in response to a request to create a named encryption key from a particular user, determining whether the particular user has sufficient privileges to create an encryption key.

74. (Currently amended) The method of claim 71, wherein the named encryption key is itself encrypted ~~to~~ by a key-encrypting key constructed from a user-supplied password.

75. (Previously presented) The method of claim 74, wherein the named encryption key is encrypted using Advanced Encryption Standard (AES) encryption.

76. (Original) The method of claim 74, wherein the user-supplied password may

comprise a hex literal.

77. (Original) The method of claim 74, wherein the user-supplied password is itself transformed into a symmetric encryption key, using a random salt, static internal data and SHA-1 hashing algorithm.

78. (Original) The method of claim 71, wherein the database system stores encrypted column data internally as variable binary (VARBINARY) data.

79. (Original) The method of claim 71, wherein columns of the database table that are not specified to be encrypted are stored in unencrypted format.

80. (Currently amended) The method of claim 71, wherein the system implements said ~~first and second statements~~ extensions as SQL extensions for creating and managing named encryption keys and for creating and managing database tables with encrypted column data.

81. (Previously presented) The method of claim 80, wherein said SQL extensions include a CREATE ENCRYPTION KEY command for creating a named encryption key.

82. (Original) The method of claim 81, wherein said CREATE ENCRYPTION KEY command includes attributes specifying an encryption key name and a user-supplied password.

83. (Original) The method of claim 80, wherein said SQL extensions include a CREATE TABLE command having an attribute that allows specification of at least one column to be encrypted.

84. (Original) The method of claim 83, wherein said CREATE TABLE command syntax includes attributes specifying a table name, one or more columns to be encrypted, and an encryption key name.

85. (Original) The method of claim 71, wherein said second query language statement includes a request specifying alteration of a previously-created table so as to encrypt particular column data.

86. (Previously presented) The method of claim 71, wherein a user subsequently requiring use of the encrypted column data must provide a user-supplied password for unlocking the named encryption key for the particular column.

87. (Currently amended) The method of claim 71, further comprising:

receiving ~~an SQL~~ a query language statement specifying creation of a default key encryption password.

88. (Currently amended) The method of claim 87, wherein the ~~SQL~~ query language statement specifying creation of a default key encryption password specifies a default password value that is encrypted by a system stored procedure, for storage in a system table of a particular database.

89. (Currently amended) The method of claim 71, further comprising:

receiving ~~an SQL~~ a query language statement specifying creation of an encryption keypair.

90. (Currently amended) The method of claim 89, wherein the ~~SQL~~ query language statement specifying creation of an encryption keypair comprises a CREATE ENCRYPTION KEYPAIR command.

91. (Original) The method of claim 90, wherein the CREATE ENCRYPTION KEYPAIR command includes:

```
CREATE ENCRYPTION KEYPAIR keypairname
  [FOR algorithm]
  [WITH [KEYLENGTH keysize]
```

[PASSWD passphrase | LOGIN_PASSWD]
as its syntax.

92. (Currently amended) The method of claim 71, further comprising:
receiving ~~an SQL~~ a query language statement specifying alteration of a particular named encryption key or keypair.

93. (Currently amended) The method of claim 71, further comprising:
receiving ~~an SQL~~ a query language statement specifying dropping a particular named encryption key or keypair.

94. (Currently amended) The method of claim 71, further comprising:
receiving ~~an SQL~~ a query language statement granting rights to a particular named encryption key or keypair.

95. (Currently amended) The method of claim 94, further comprising:
receiving ~~an SQL~~ a query language statement revoking said rights that have been granted to a particular named encryption key or keypair.

96. (Currently amended) The method of claim 94, wherein ~~the~~ said rights granted for the particular named encryption key or keypair comprise SELECT query execution rights, for selecting encrypted data.

97. (Currently amended) The method of claim 94, wherein ~~the~~ said rights granted for the particular named encryption key or keypair comprise ALTER query execution rights, for altering the encryption key or keypair.

98. (Original) A computer-readable medium having processor-executable instructions for performing the method of claim 71.

99. (Original) A downloadable set of processor-executable instructions for

performing the method of claim 71.